

Sequence Alignment: Linear Space

Q. Can we avoid using quadratic **space**?

Easy. Optimal **value** in $O(m + n)$ space and $O(mn)$ time.

- Compute $\text{OPT}(i, \cdot)$ from $\text{OPT}(i-1, \cdot)$.
- No longer a simple way to recover alignment itself.

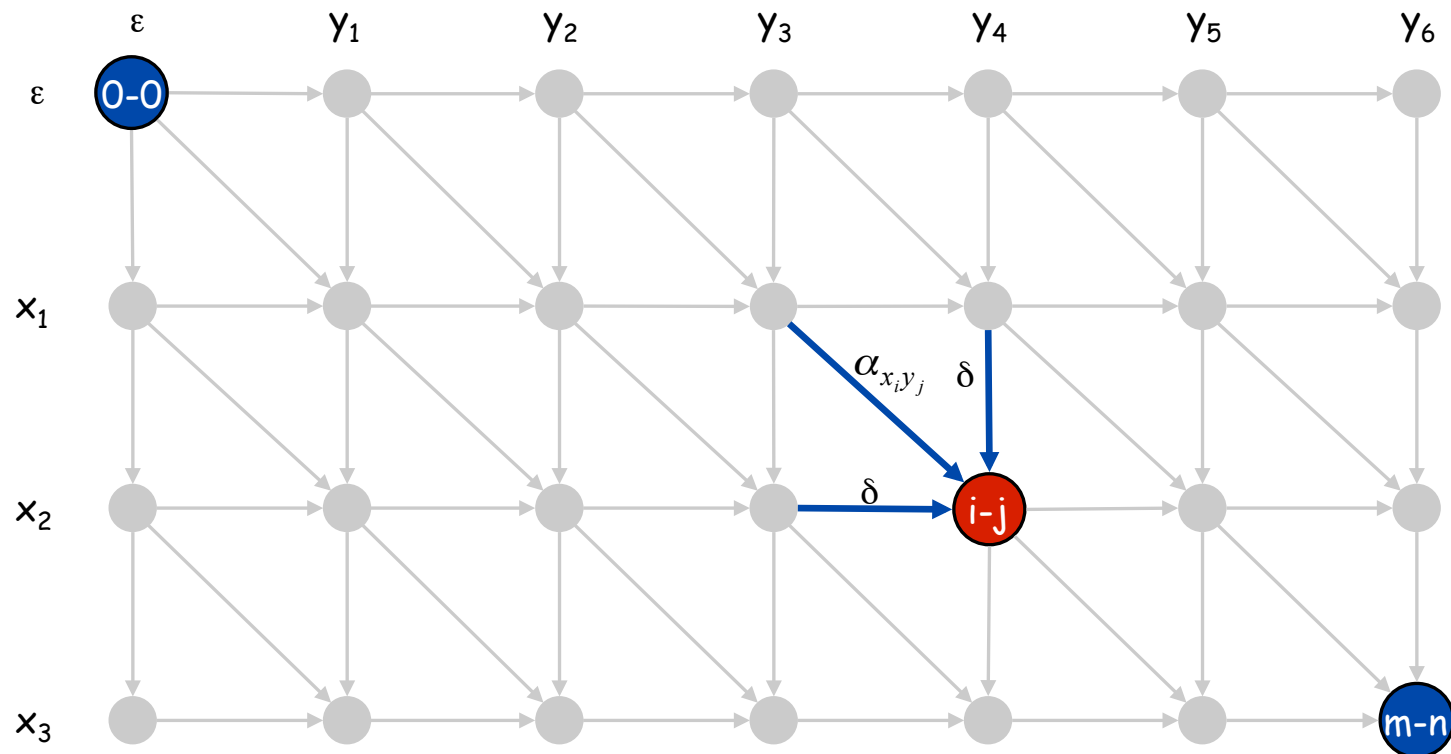
Theorem. [Hirschberg 1975] Optimal **alignment** in $O(m + n)$ space and $O(mn)$ time.

- Clever combination of divide-and-conquer and dynamic programming.
- Inspired by idea of Savitch from complexity theory.

Sequence Alignment: Linear Space

Edit distance graph.

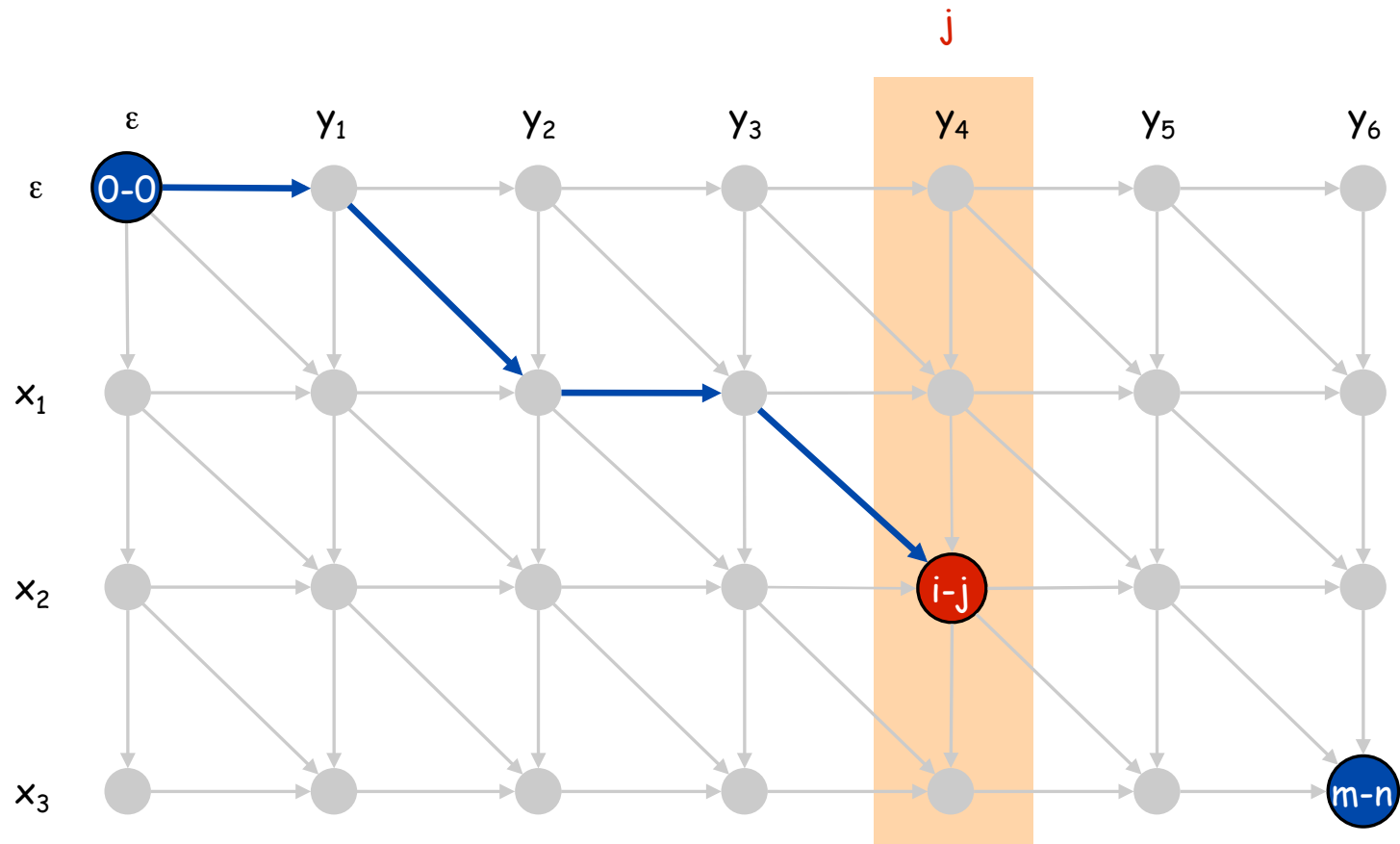
- Let $f(i, j)$ be shortest path from $(0,0)$ to (i, j) .
- Observation: $f(i, j) = \text{OPT}(i, j)$.



Sequence Alignment: Linear Space

Edit distance graph.

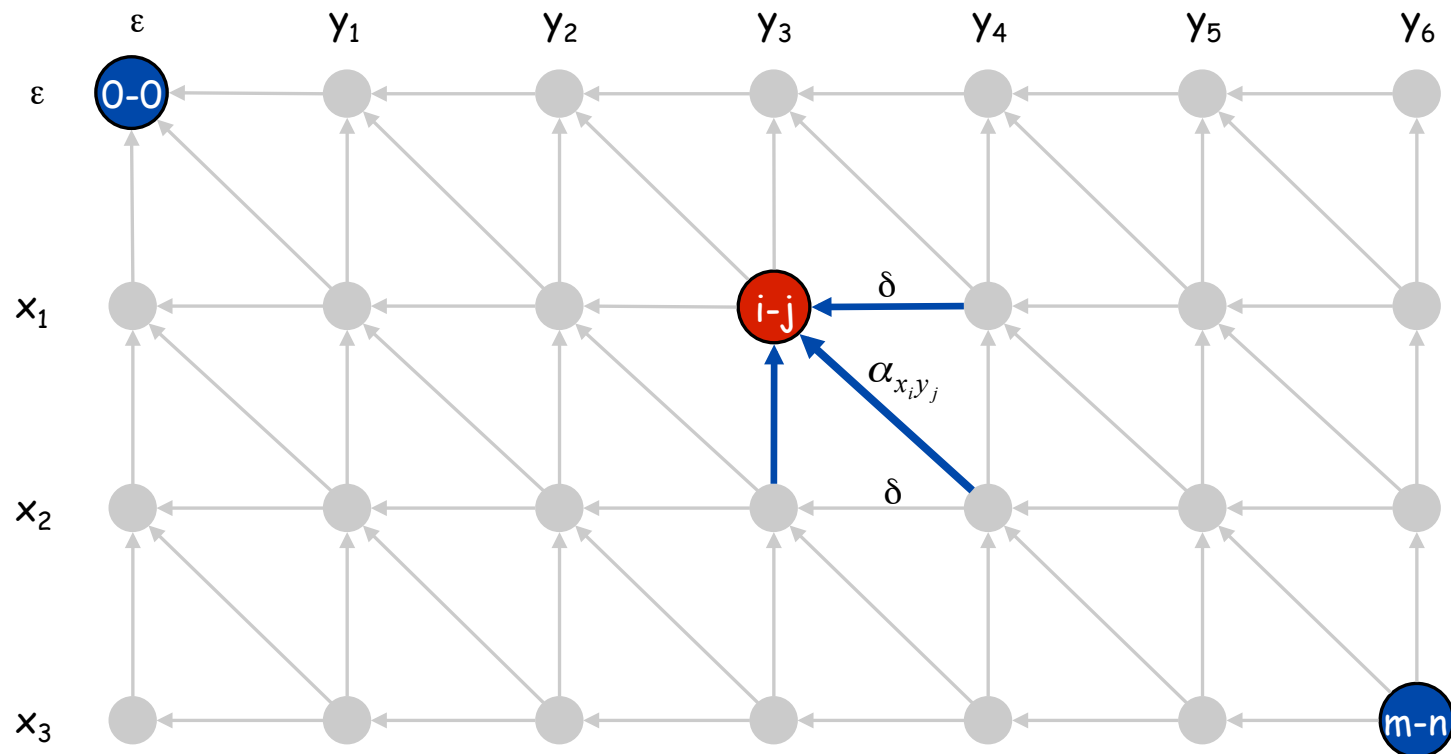
- Let $f(i, j)$ be shortest path from $(0,0)$ to (i, j) .
- Can compute $f(\cdot, j)$ for any j in $O(mn)$ time and $O(m + n)$ space.



Sequence Alignment: Linear Space

Edit distance graph.

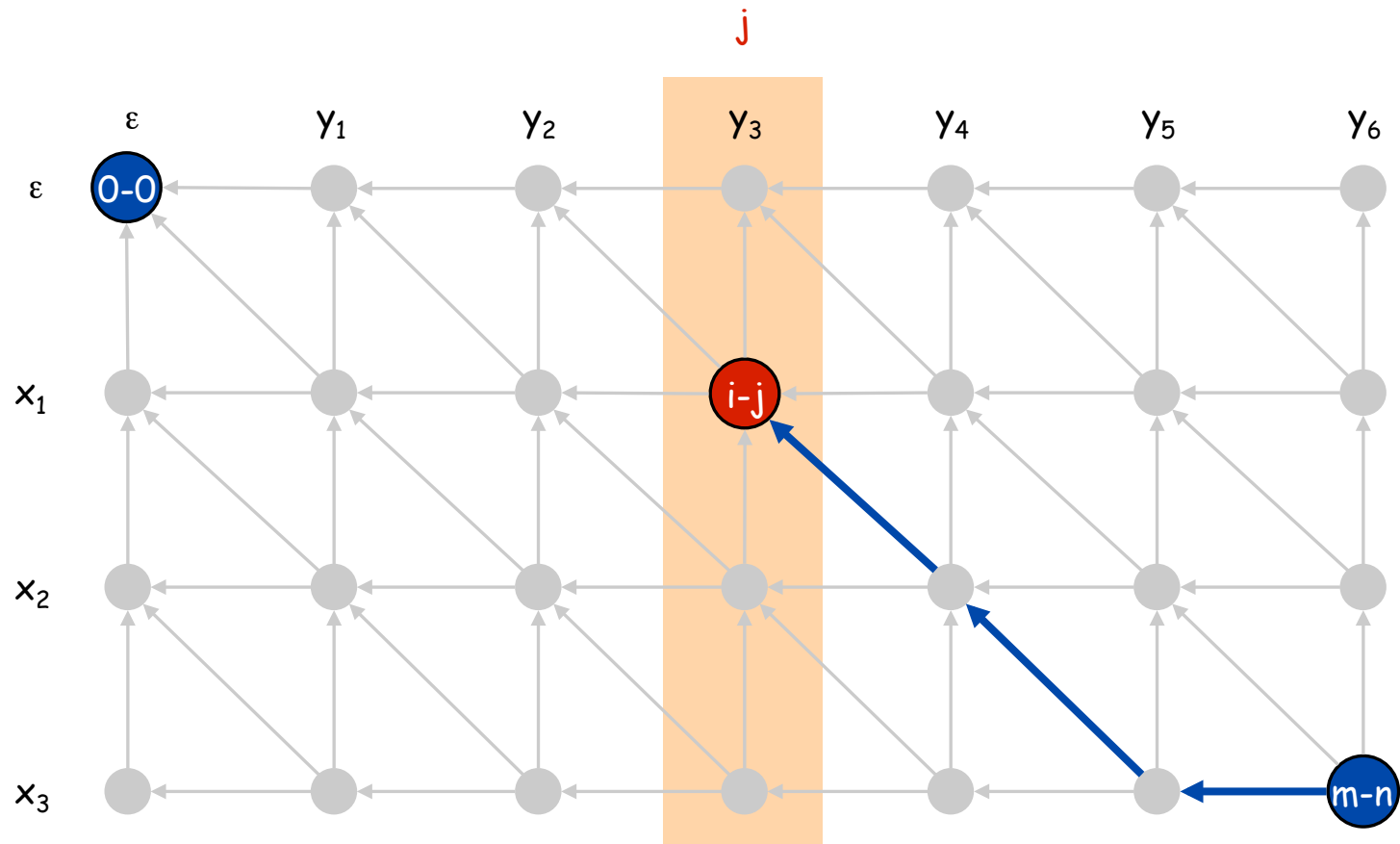
- Let $g(i, j)$ be shortest path from (i, j) to (m, n) .
- Can compute by reversing the edge orientations and inverting the roles of $(0, 0)$ and (m, n)



Sequence Alignment: Linear Space

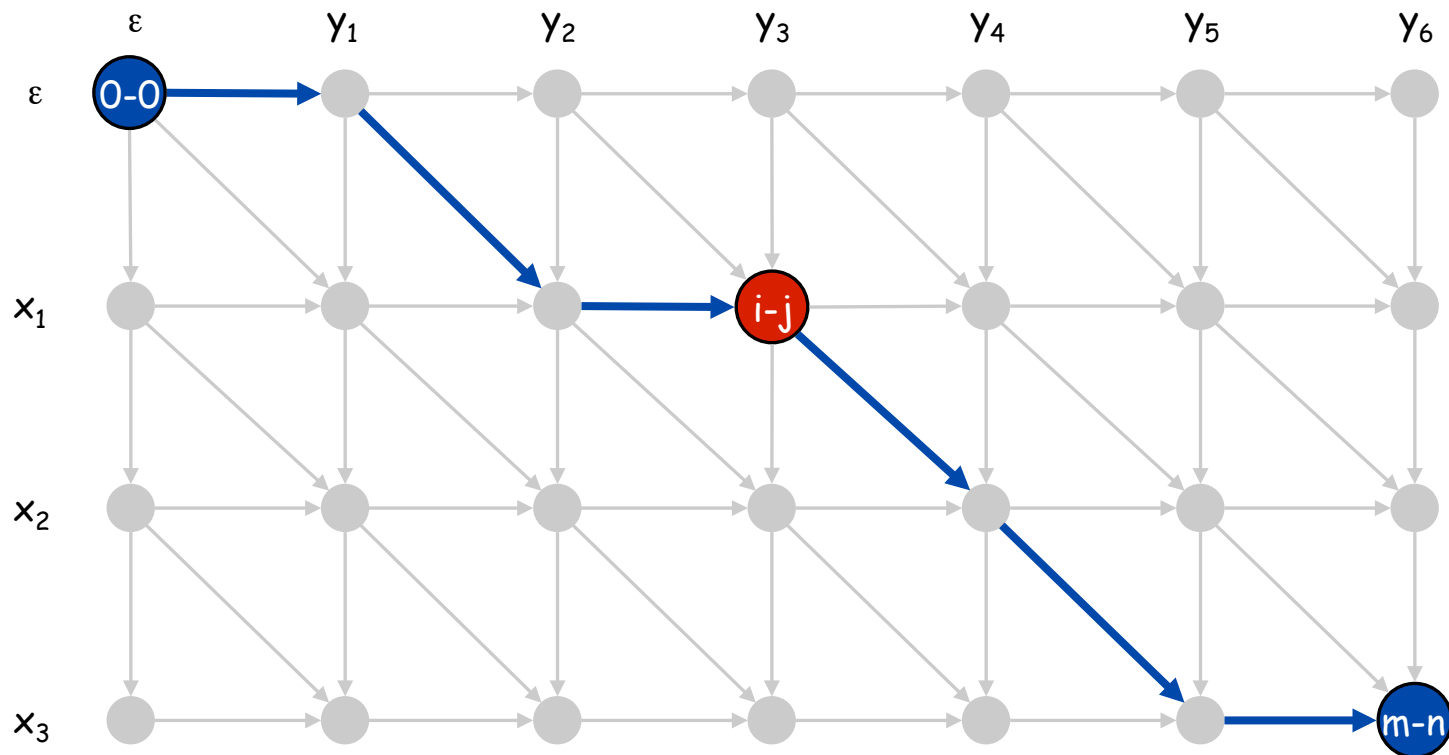
Edit distance graph.

- Let $g(i, j)$ be shortest path from (i, j) to (m, n) .
- Can compute $g(\cdot, j)$ for any j in $O(mn)$ time and $O(m + n)$ space.



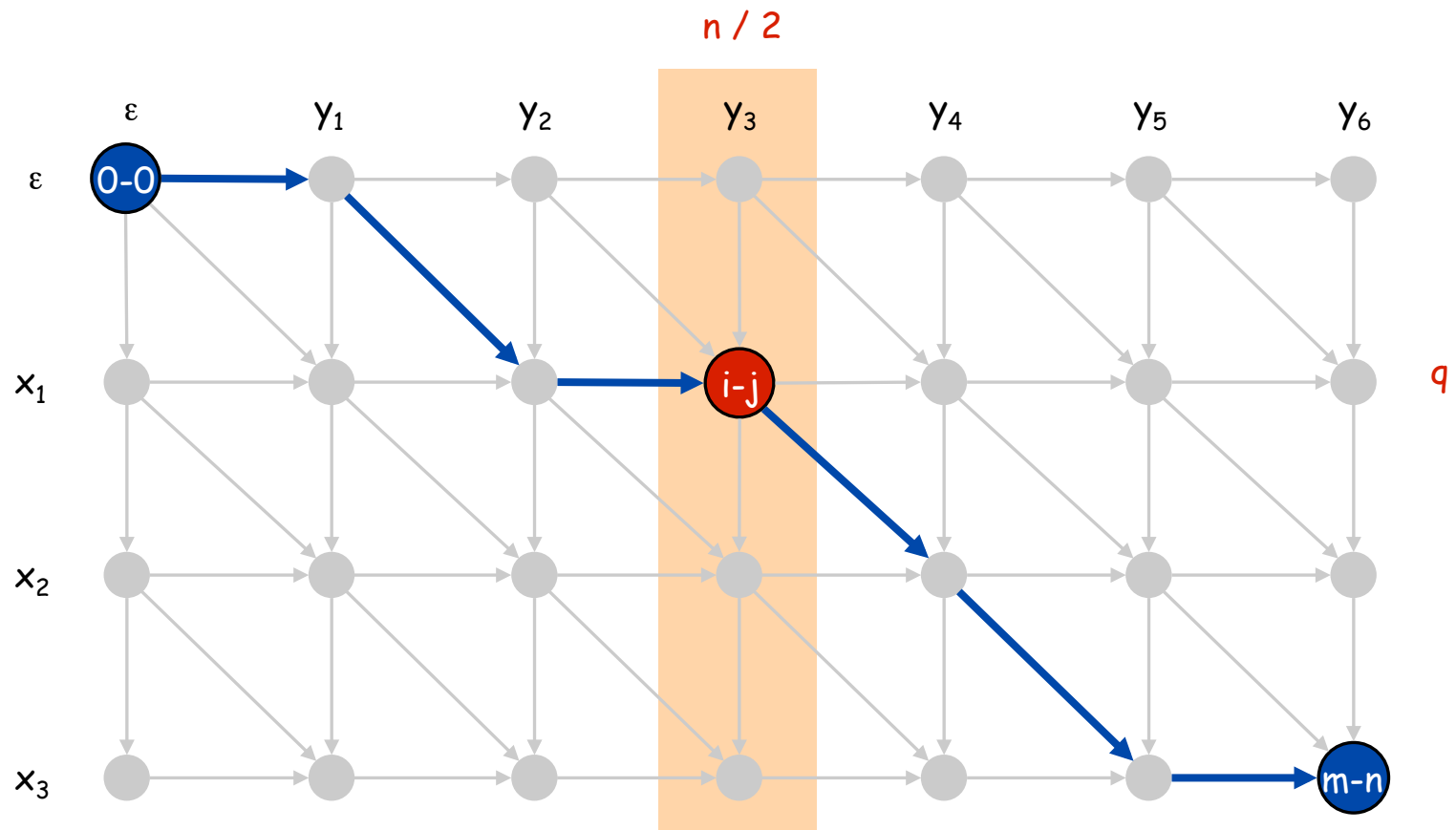
Sequence Alignment: Linear Space

Observation 1. The cost of the shortest path that uses (i, j) is $f(i, j) + g(i, j)$.



Sequence Alignment: Linear Space

Observation 2. let q be an index that minimizes $f(q, n/2) + g(q, n/2)$. Then, the shortest path from $(0, 0)$ to (m, n) uses $(q, n/2)$.

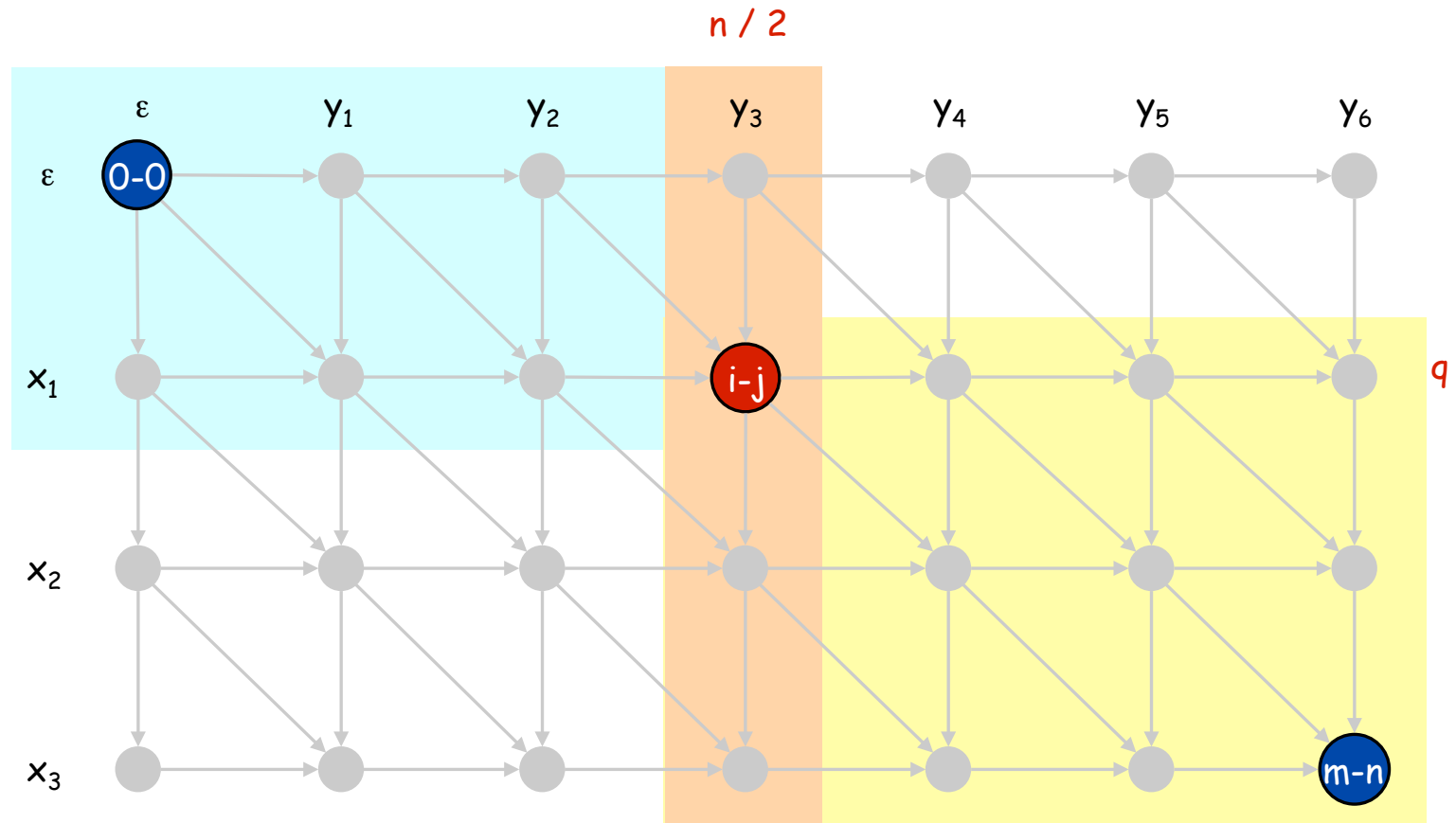


Sequence Alignment: Linear Space

Divide: find index q that minimizes $f(q, n/2) + g(q, n/2)$ using DP.

- Align x_q and $y_{n/2}$.

Conquer: recursively compute optimal alignment in each piece.



Sequence Alignment: Running Time Analysis Warmup

Theorem. Let $T(m, n)$ = max running time of algorithm on strings of length at most m and n . $T(m, n) = O(mn \log n)$.

$$T(m, n) \leq 2T(m, n/2) + O(mn) \Rightarrow T(m, n) = O(mn \log n)$$

Remark. Analysis is not tight because two sub-problems are of size $(q, n/2)$ and $(m - q, n/2)$. In next slide, we save $\log n$ factor.

Sequence Alignment: Running Time Analysis

Theorem. Let $T(m, n)$ = max running time of algorithm on strings of length m and n . $T(m, n) = O(mn)$.

Pf. (by induction on n)

- $O(mn)$ time to compute $f(\cdot, n/2)$ and $g(\cdot, n/2)$ and find index q .
- $T(q, n/2) + T(m - q, n/2)$ time for two recursive calls.
- Choose constant c so that:

$$T(m, 2) \leq cm$$

$$T(2, n) \leq cn$$

$$T(m, n) \leq cmn + T(q, n/2) + T(m - q, n/2)$$

- Base cases: $m = 2$ or $n = 2$.
- Inductive hypothesis: $T(m, n) \leq 2cmn$.

$$\begin{aligned} T(m, n) &\leq T(q, n/2) + T(m - q, n/2) + cmn \\ &\leq 2cq n/2 + 2c(m - q)n/2 + cmn \\ &= cq n + cmn - cq n + cmn \\ &= 2cmn \end{aligned}$$